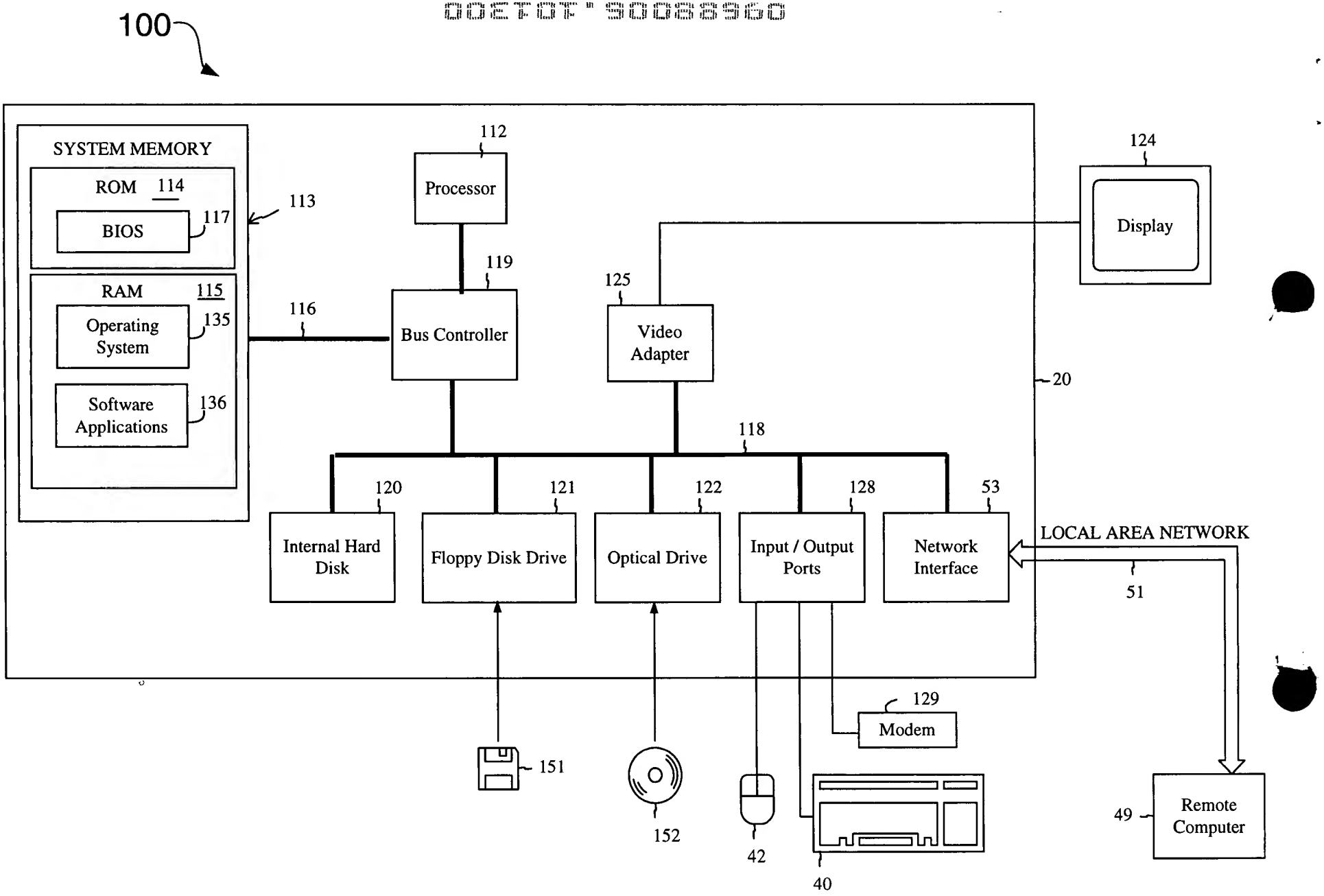

FIG. 1

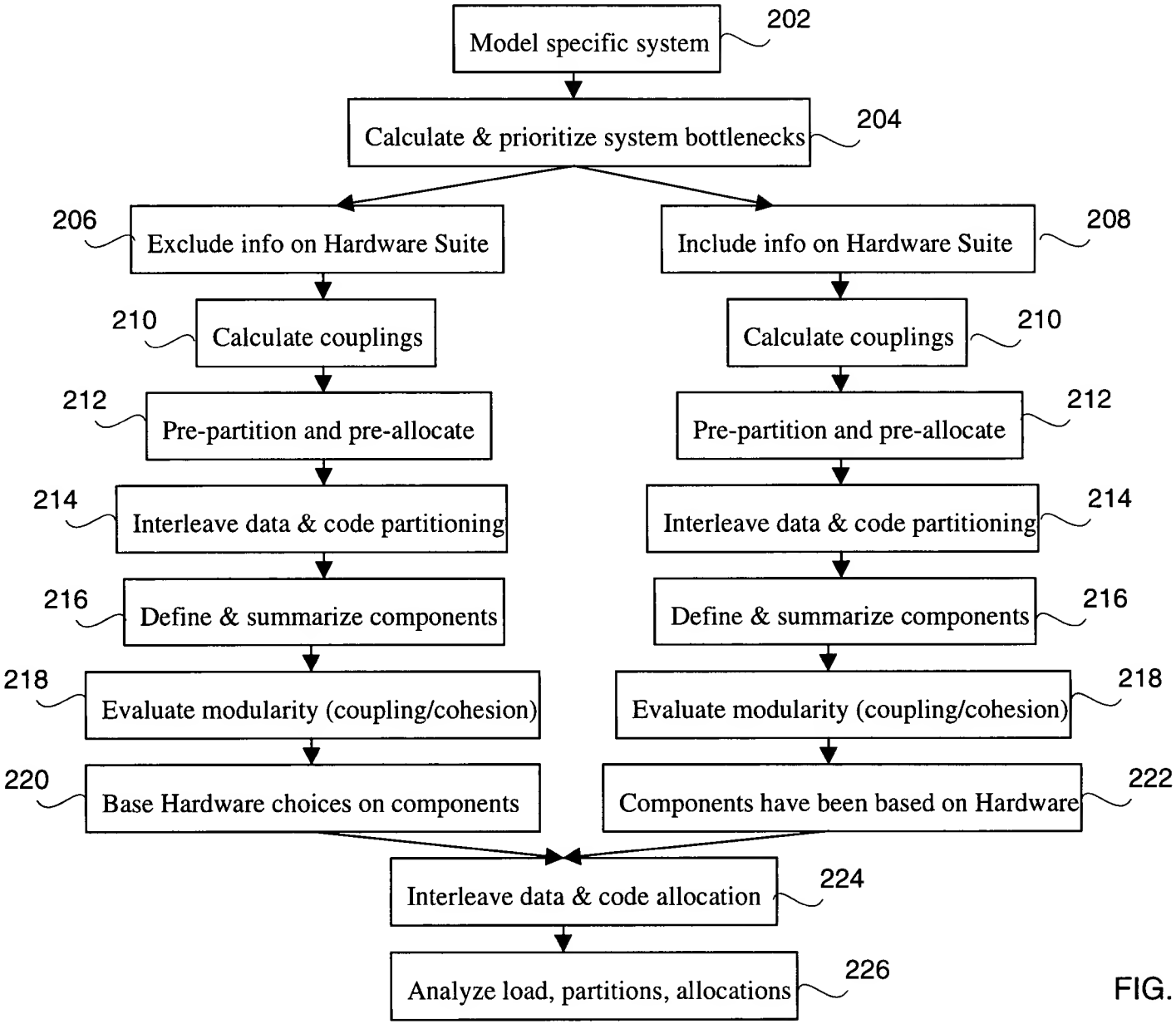
Model specific system
202
Calculate & prioritize system bottlenecks
204
206
Exclude info on Hardware Suite
208
Include info on Hardware Suite
210
Calculate couplings
210
Calculate couplings
212
Pre-partition and pre-allocate
212
Pre-partition and pre-allocate
214
Interleave data & code partitioning
214
Interleave data & code partitioning
216
Define & summarize components
216
Define & summarize components
218
Evaluate modularity (coupling/cohesion)
218
Evaluate modularity (coupling/cohesion)
220
Base Hardware choices on components
222
Components have been based on Hardware
Interleave data & code allocation
224
Analyze load, partitions, allocations
226

FIG. 2

In the following algorithm, Step 1 calculates the appropriate execution time and processor utilization, Step 2.4 then calculates the control coupling as a combination of calculating: in Step 2.1 the latency, Step 2.2 the timing strength, and Step 2.3 the frequency strength. Step 2.5 then stores this new information into the coupling matrix.

1. For each T_i, do:
   - 1.1 Choose P_l from {default or each available type}
   - 1.2 Calculate execution time $T_i[ET]$ based on the chosen P_l (single or multiple)
   - 1.3 Calculate processor utilization $T_i[PU]$ based on the chosen P_l (single or multiple)
2. For each T_i, loop through all tasks $(T_n \mid T_i \rightarrow T_n)$,
   - 2.1 Calculate the latency for each task coupling, $TT_{i \rightarrow n}[LT]$, (assumes pipelined structure and uniform bus) as:
     $TT_{i \rightarrow n}[LT]$ = *communication constant*
   - 2.2 Rank coupling latency, $TT_{i \rightarrow n}[LT]$, against the Timing Step Function to calculate the timing strength for this coupling, $TT_{i \rightarrow n}[TS]$
   - 2.3 Rank coupling frequency, $TT_{i \rightarrow n}[FQ]$, against the Frequency Step Function to calculate the frequency strength for this coupling, $TT_{i \rightarrow n}[FS]$
   - 2.4 Calculate the control coupling as,
     $TT_{i \rightarrow n}[CC] = \max\{TT_{i \rightarrow n}[TS], TT_{i \rightarrow n}[FS]\} + \{1, \text{if both } TS \text{ \& } FS > 5 \mid 0, \text{else}\}$
   - 2.5 Add the constraint $TT_{i \rightarrow n}$ [CC] to the coupling matrix

End

6.3.3.2 Data Couplings

The system's data flow contains both task input and output, represented individually to support analysis of the data couplings, and subsequent partitioning and allocation decisions, prior to determining which data blocks are best implemented into messages, shared memory, or databases.

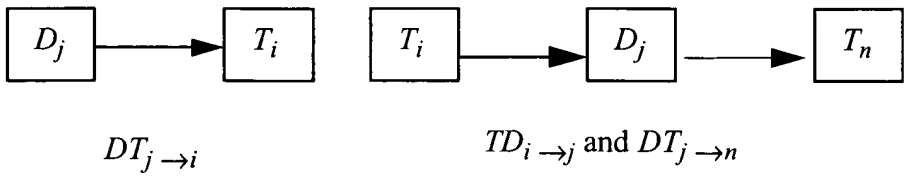

Figure 3 Input vs. Output Data Block Access

```
        2.1.1Within this sorted list, for each set of triples where
             max[processor.ratio, memory.ratio] is equal,
             2.1.1.1     sort this subset of triples by
                  min[processor.ratio, memory.ratio].
End
```

6.3.3 Calculate Couplings

The following formulas calculate the control, data, and peripheral couplings from the domain model.

The structure of the communication (inter-task and task-data) is assumed to be a pipelined message, which is used to calculate the latency in the couplings. This basic structure represents both control and data flow, with data flow containing a volume of data, while control flow consumes minimal bandwidth. Modifications of this algorithm easily extend to include the other remote procedure call and messaging structures, at any level of nesting.

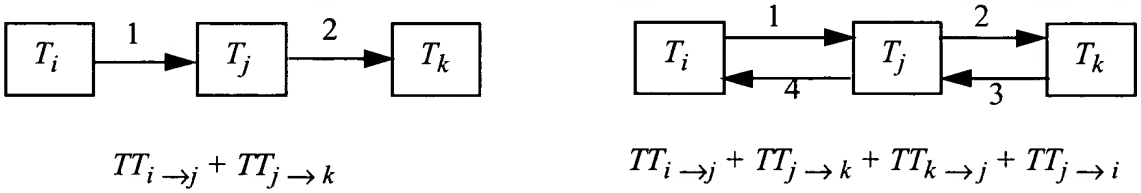

FIG. 3A

~~Figure 2~~ Pipelined vs. Nested Communication

For example, the impact on data flow latency of different communication structures, such as rendezvous vs. direct schedule, adds the following calculations:

```
1.  CASE IPC is
    1.1    Rendezvous Output
           {TD_i→j}[LT] = T_i[RST] - D_j[RR]

    1.2    Rendezvous Input
           {DT_j→i}[LT] = T_i[RST] - D_j[AT]

    1.3    Direct Schedule Output
           {TD_i→j}[LT] = communication constant

    1.4    Direct Schedule Input
           {DT_j→i}[LT] = communication constant

    End Case
```

section 6.3.3.3 Peripheral Couplings discusses the peripheral I/O specifics.

6.3.3.1 Control Couplings

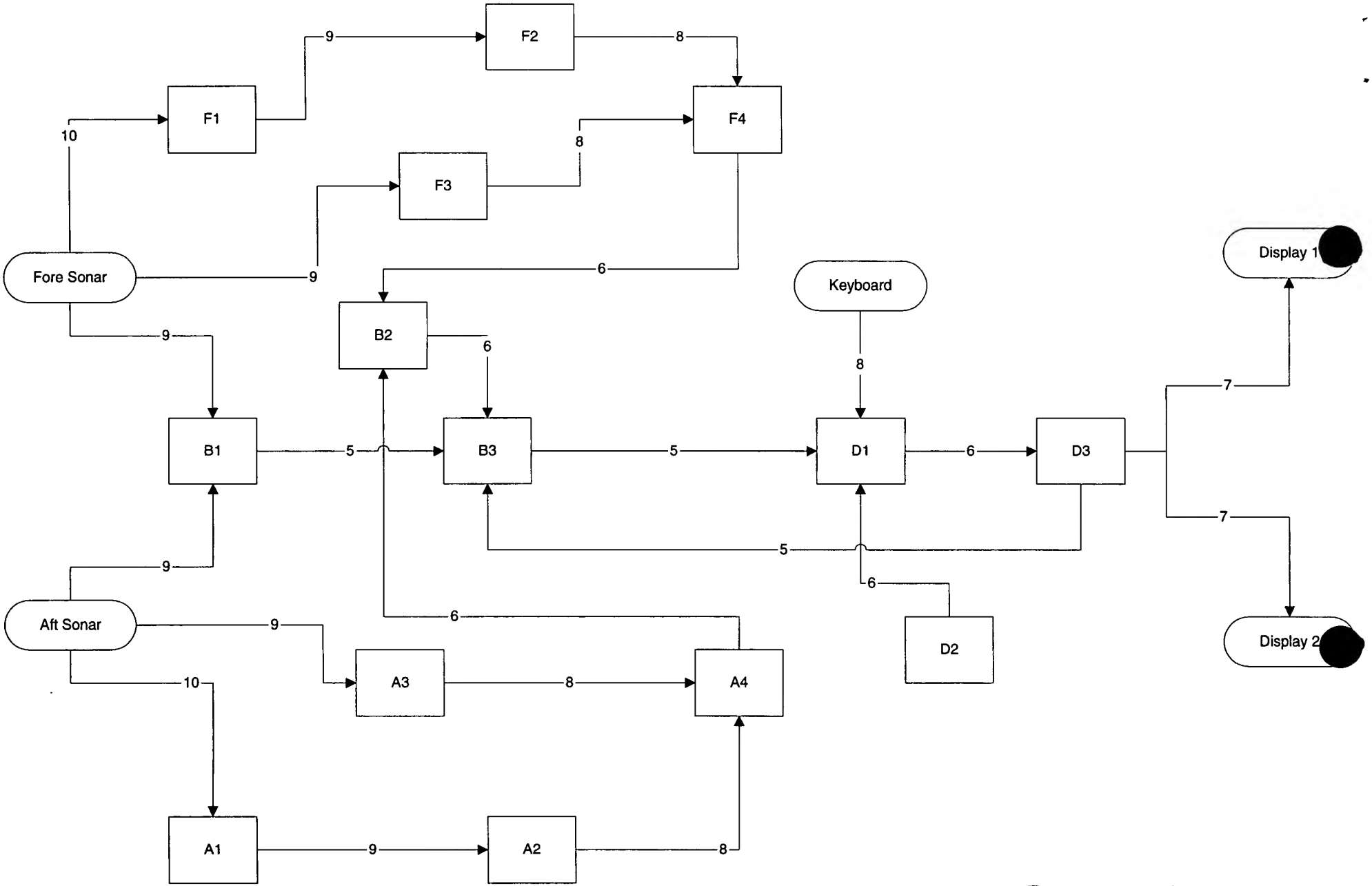
F2
9
8
F1
F4
10
8
F3
Fore Sonar
9
6
Keyboard
Display 1
B2
6
9
8
7
B1
5
B3
5
D1
6
D3
9
7
5
6
Aft Sonar
9
6
D2
Display 2
A3
8
A4
10
A1
9
A2
8

FIG. 4

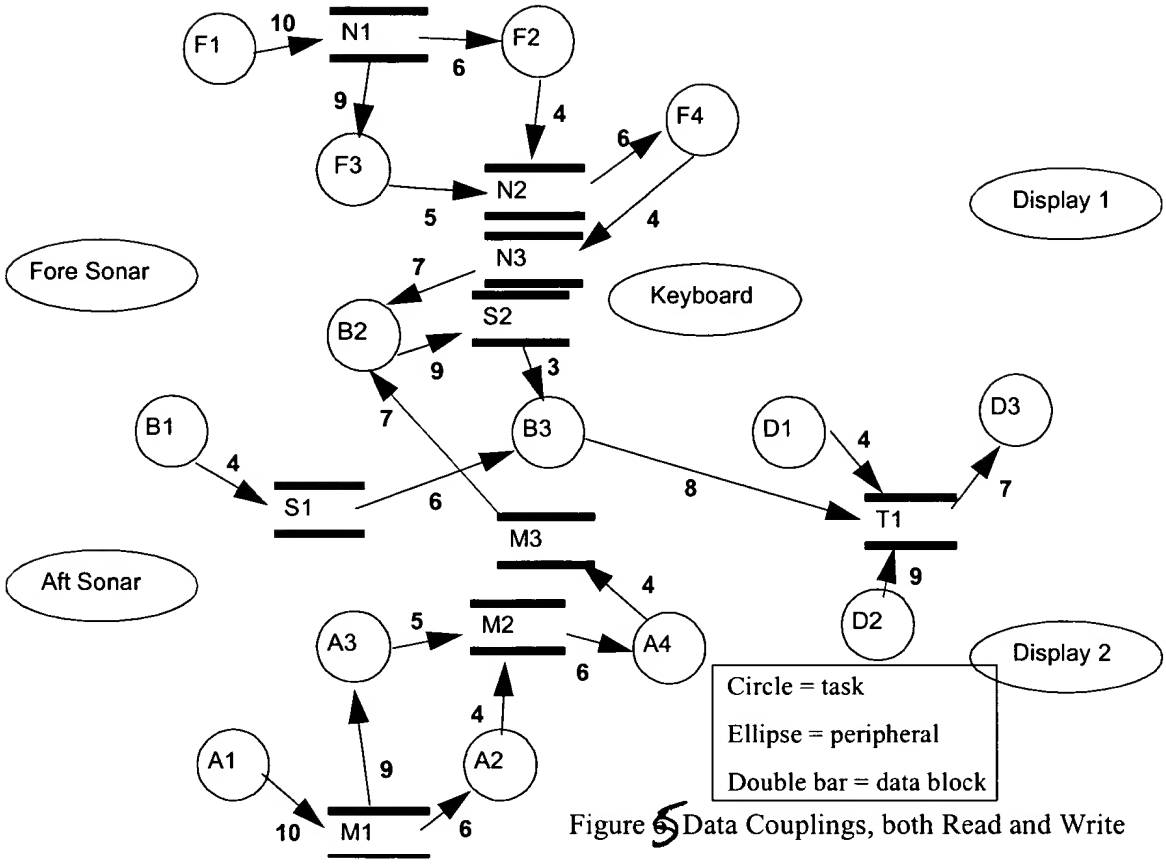

Figure 5 Data Couplings, both Read and Write

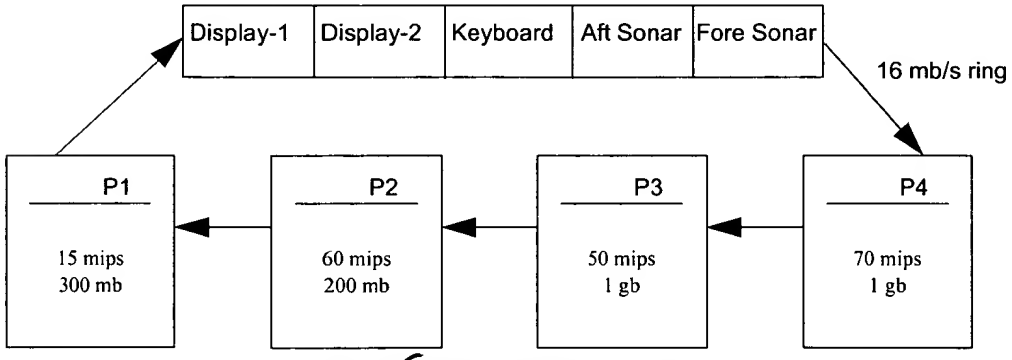

Figure 6 Initial Hardware Suite

7.5 Couplings

Table 13 shows the triples for this example, implemented as a set of three columns (control, data, and peripheral), and ten rows (values 1 - 10) representing the range of coupling strengths.

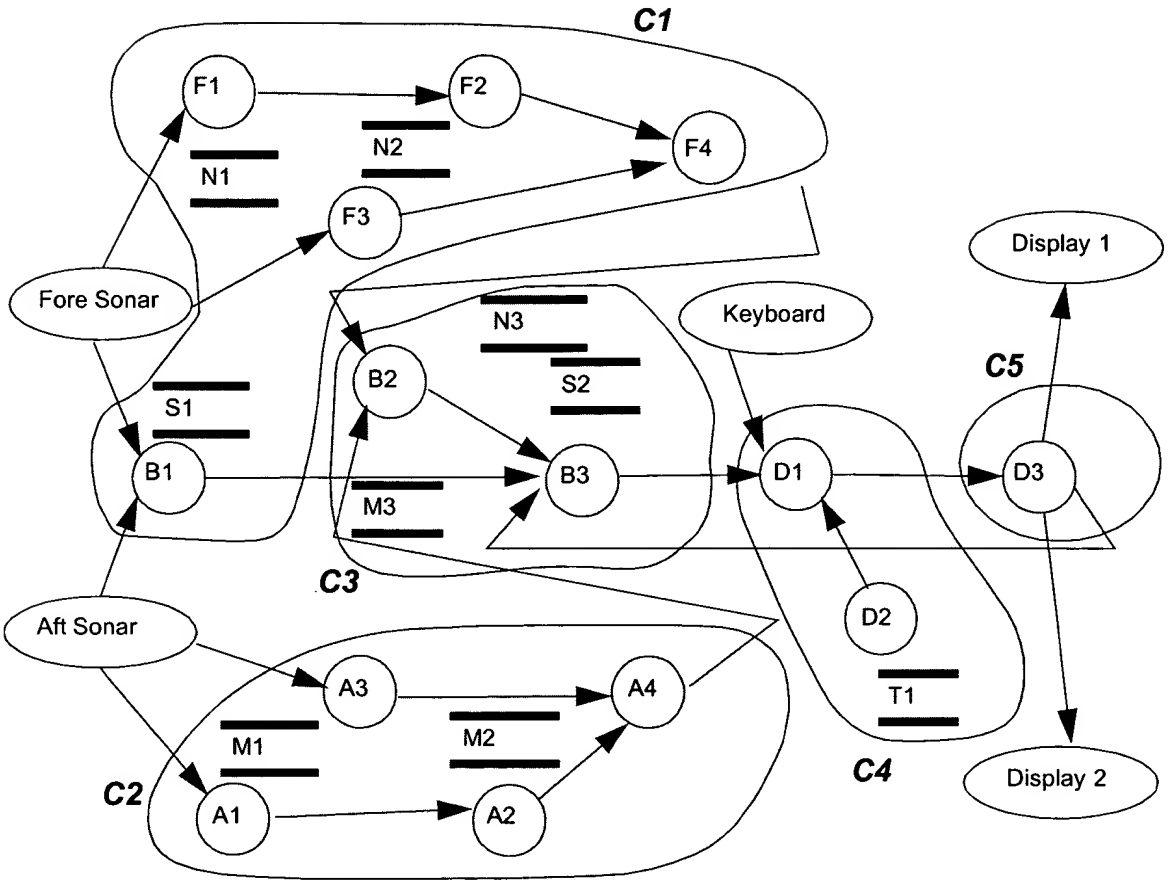

Figure 7 Partitioned Components, *Cn*

7.7 Component Totals

The results for this example system's initial estimates are shown in Figure 9.

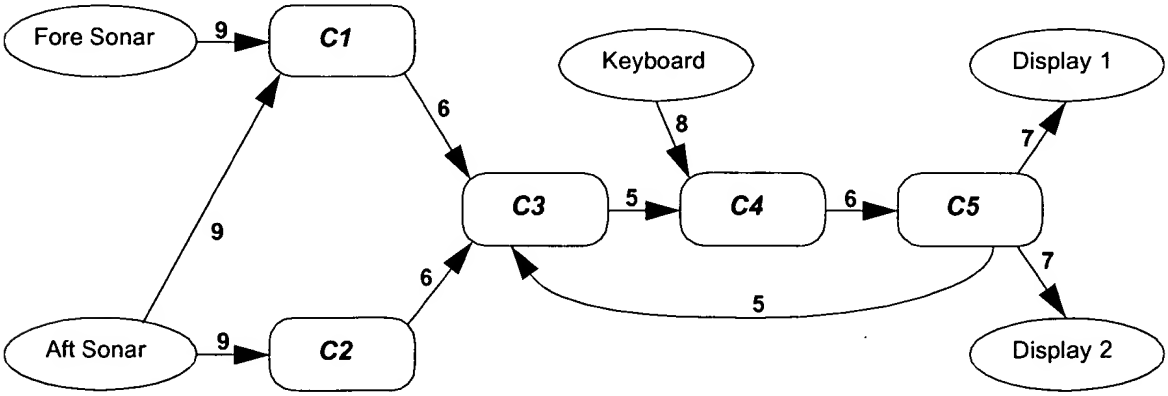

Figure 8 Summarized Components

7.8 Allocation Results

The resulting allocation for the example's original estimates is shown in Figure 10.

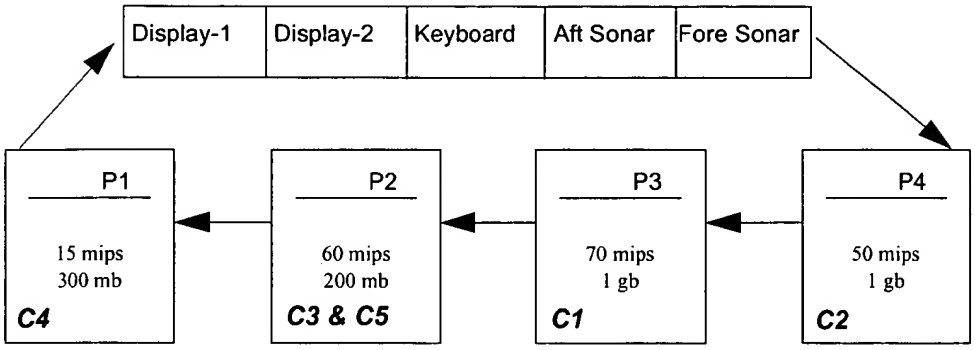

Figure 10 Example Allocation

The resulting worst case loads are shown below in Table 19.

Table 19 Initial Platform Loads

Platform *(Component)*	Processor Capacity	Processor Utilization	Memory Capacity	Memory Utilization
P1 *(C4)*	15 mips	68%	300 mb	84%
P2 *(C3 &C5)*	60 mips	93%	200 mb	88%
P3 *(C1)*	70 mips	89%	1 gb	90%
P4 *(C2)*	50 mips	90%	1 gb	80%

Several architectural tradeoffs again must be made during allocation. For example, the Components C1 and C2 fill up Processors P3 and P4, respectively. Had these two processors exchanged locations on the ring bus with P1 and P2, there would have been a conflict between allocating C1 and C2 into processors that had sufficient capacity, versus the extra network traffic caused by the additional hops from the input sensors. Also, C1 requires more CPU capacity than C2, due to the inclusion of Task B1. Platform Loads

Table 20 shows the resultant loads caused by modifying the platform capacities to combine the original first and second platforms (P1 and P2 are now Px). As shown, Px contains almost 30% more memory than needed for the worst case load, while the CPU loads also have significant excess capacity. These worst case values are calculated to support the required response time, so this may indicate an opportunity to save on hardware costs.

Table 20 Modified Platform Loads

Platform *(Component)*	Processor Capacity	Processor Utilization	Memory Capacity	Memory Utilization
Px *(C3, C4 & C5)*	80 mips	82%	600 mb	71%
P3 *(C1)*	70 mips	86%	1 gb	80%
P4 *(C2)*	50 mips	90%	1 gb	90%

Figure 11 describes one change that might result from the understanding of the peripheral bottleneck described in section 7.5 Couplings. Now the peripherals are re-located from a single place on a ring bus to direct processor connections. In this analysis, the initial configuration was used to decide that the peripherals needed to be moved. After changing the

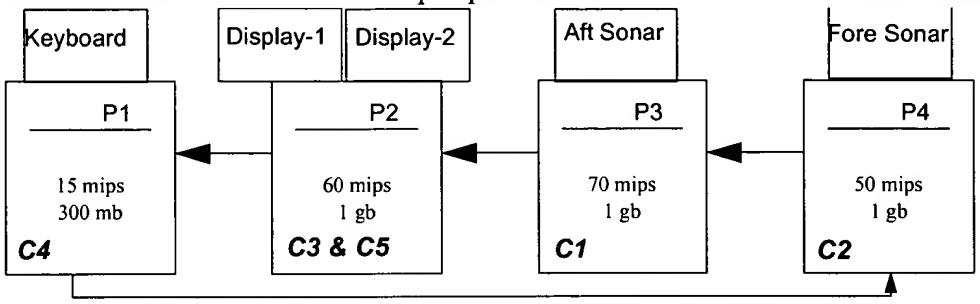

Figure 10 Modified Hardware Suite and Resulting Allocation

model to reflect the changed hardware suite, the process then re-evaluated and generated the configuration shown. This confirmed that the change in the hardware suite was appropriate, and illustrates the recursive impact that the partitioning and allocation tradeoffs have.